

# Algoritmos y Estructuras de Datos II

## Segundo recuperatorio

Sábado 11 de diciembre de 2021

### Aclaraciones

- El examen es individual en todas sus etapas, y a libro abierto.
- Se responderán consultas por Discord, sólo de interpretación del enunciado, a través del canal ‘Mesa de Docentes’ de la categoría PARCIALES.
- Cualquier aclaración sobre el enunciado se realizará a través del canal ‘anuncios-evaluaciones’.
- El parcial durará 4 horas, de 9 a 13hs, y la entrega se realizará a través del Campus hasta las 13:15hs, en forma estricta. Se desestimarán entregas fuera de tiempo, sin excepciones.
- El ejercicio se calificará con **P**erfecto, **A**probado (que puede ir con ‘—’ o ‘?’), **R**egular, o **I**nsuficiente.
- Para aprobar la cursada se deben tener todos los ejercicios aprobados de todos los parciales.
- Los ejercicios con **R** o **I** se recuperarán por separado al final del cuatrimestre.
- Sólo se podrá tener un ejercicio con ‘A?’ en la cursada, si no, deberán recuperarse los que hagan falta.

### Ej. 1. Elección de estructuras

Luego de relevar las necesidades del mercado de cursos en línea, el proyecto *Courseasy* está listo para avanzar en el diseño de la plataforma que llevará el mismo nombre.

Se espera que *Courseasy* disponga de un catálogo de cursos *online*, tanto gratuitos como pagos, que los usuarios podrán ofrecer a través de la plataforma y, también, estos podrán realizar los cursos que deseen.

Cada curso de la plataforma contará con la siguiente información: Id del curso (un número natural no acotado, para el cual disponen de una función que les da un id disponible<sup>1</sup>), nombre (con máxima longitud acotada), área temática principal (con máxima longitud acotada, e.g., ‘pintura’, ‘fotografía’ y ‘programación’) de un listado predefinido de áreas, cupo, precio y docente a cargo (tiene que ser usuario de la plataforma).

Además, en la plataforma deberán manejarse las inscripciones (y las desinscripciones) a los cursos, además de poder realizar búsquedas de los cursos de acuerdo a diferentes criterios, además de identificar aquellos cursos que aún tiene cupo disponible.

A continuación se tiene una especificación parcial de este problema.

**TAD** CURSO es NAT

**TAD** DOCENTE es NAT

**TAD** ESTUDIANTE es NAT

**TAD** DENOMINACION es STRING

**TAD** AREA es STRING

**TAD** PRECIO es NAT

**TAD** CUPO es NAT

**TAD** USUARIO es NAT

<sup>1</sup>La función se llama NUEVOID y no necesariamente devuelve valores contiguos si se la llama dos veces consecutivas.

**TAD COURSEASY****observadores básicos**

cursos	: courseasy	$\longrightarrow$ conj(curso)	
nombre	: courseasy $c \times$ curso $k$	$\longrightarrow$ denominacion	$\{k \in \text{cursos}(c)\}$
cupo	: courseasy $c \times$ curso $k$	$\longrightarrow$ cupo	$\{k \in \text{cursos}(c)\}$
area	: courseasy $c \times$ curso $k$	$\longrightarrow$ área	$\{k \in \text{cursos}(c)\}$
docente	: courseasy $c \times$ curso $k$	$\longrightarrow$ docente	$\{k \in \text{cursos}(c)\}$
precio	: courseasy $c \times$ curso $k$	$\longrightarrow$ precio	$\{k \in \text{cursos}(c)\}$
inscriptosEnCurso	: courseasy $c \times$ curso $k$	$\longrightarrow$ conj(estudiante)	$\{k \in \text{cursos}(c)\}$
estáDisponible?	: courseasy $c \times$ curso $k$	$\longrightarrow$ bool	$\{k \in \text{cursos}(c)\}$

**generadores**

inicia	:	$\longrightarrow$ courseasy	
agregarCurso	: courseasy $c \times$ curso $k \times$ denominacion $d$	$\longrightarrow$ courseasy	
	$\times$ área $\times$ docente $\times$ precio $\times$ cupo $q$		$\{k \notin \text{cursos}(c) \wedge \neg \text{vacía?}(d) \wedge q > 0\}$
aumentarCupo	: courseasy $c \times$ curso $k \times$ cupo $q$	$\longrightarrow$ courseasy	
			$\{k \in \text{cursos}(c) \wedge q > 0\}$
inscribirACurso	: courseasy $c \times$ curso $k \times$ estudiante $e$	$\longrightarrow$ courseasy	
			$\{k \in \text{cursos}(c) \wedge \text{estáDisponible?}(c, k) \wedge e \notin \text{inscriptosEnCurso}(c, k)\}$
cerrarInscripción	: courseasy $c \times$ curso $k$	$\longrightarrow$ courseasy	
			$\{k \in \text{cursos}(c) \wedge \text{estáDisponible?}(c, k)\}$

**Fin TAD**

Dadas las siguientes operaciones, que son sólo algunas de las que posee el módulo, y de acuerdo a las complejidades temporales de peor caso indicadas, donde  $n$  es la cantidad de cursos actuales (sin contar eliminados),  $u$  es la cantidad de usuarios, e  $i$  es la cantidad de inscriptos a un curso determinado:

- **PUBLICARCURSO**(**inout** sistema: courseasy, **in** nombre: string, **in** áreaTemática: string, **in** docenteACargo: usuario, **in** precio: nat, **in** cupo: nat)  $\longrightarrow$  idCurso: nat  
**{Pre: Ninguna}**  
Agrega un curso al sistema con los datos definidos.  
Complejidad:  $O(\log(n) + \log(u))$
- **ELIMINARCURSO**(**inout** sistema: courseasy, **in** idCurso: nat)  
**{Pre: El curso indicado existe en el sistema.}**  
Elimina del sistema el curso indicado.  
Complejidad:  $O(\log(n) + i)$
- **CURSOSHISTÓRICOS**(**in** sistema: courseasy)  $\longrightarrow$  conjunto(curso)  
**{Pre: Ninguna.}**  
Obtiene todos los cursos que se publicaron alguna vez en la plataforma.  
Complejidad:  $O(1)$
- **CERRARINSCRIPCIÓN**(**inout** sistema: courseasy, **in** idCurso: nat)  
**{Pre: El curso indicado existe en el sistema y tiene inscripción abierta.}**  
Cierra la inscripción del curso indicado.  
Complejidad:  $O(\log(n))$

- **AUMENTARCUPO**(**inout** sistema: courseasy, **in** idCurso: nat, **in** cantidad: nat)  
 {**Pre**: El curso indicado existe en el sistema y tiene inscripción abierta.}  
 Aumenta el cupo del curso indicado en la cantidad definida.  
Complejidad:  $O(\log(n))$
- **CURSOSDELÁREA**(**in** sistema: courseasy, **in** áreaTemática: string)  
 {**Pre**: Ninguna.}  
 Obtiene los cursos del área indicada.  
Complejidad:  $O(1)$
- **CURSOSPORDOCENTE**(**in** sistema: courseasy, **in** docente: usuario)  $\rightarrow$  conjunto(curso)  
 {**Pre**: Ninguna.}  
 Obtiene los cursos del sistema del docente indicado.  
Complejidad:  $O(\log(u))$
- **INSCRIBIRCURSO**(**inout** sistema: courseasy, **in** idCurso: nat, **in** estudiante: usuario)  
 {**Pre**: El curso indicado existe en el sistema y tiene inscripción abierta, y el usuario no está inscripto al mismo.}  
 Registra la inscripción de un usuario al curso indicado.  
Complejidad:  $O(\log(n) + \log(u))$
- **DESINSCRIBIRDECURSO**(**inout** sistema: courseasy, **in** idCurso: nat, **in** estudiante: usuario)  
 {**Pre**: El curso indicado existe en el sistema, y el usuario ya está inscripto al mismo.}  
 Elimina la inscripción de un usuario al curso indicado.  
Complejidad:  $O(\log(n) + i)$
- **INSCRIPCIONES**(**in** sistema: courseasy, **in** idCurso: nat)  $\rightarrow$  conjunto(usuario)  
 {**Pre**: El curso indicado está en el sistema.}  
 Obtiene los usuarios inscriptos al curso indicado.  
Complejidad:  $O(\log(n))$
- **INSCRIPCIONESDEUSUARIO**(**in** sistema: courseasy, **in** estudiante: usuario)  $\rightarrow$  conjunto(curso)  
 {**Pre**: El usuario existe en el sistema.}  
 Obtiene los cursos a los que está inscripto el estudiante indicado.  
Complejidad:  $O(\log(u))$

Se pide:

- a) Plantear la estructura de representación del módulo COURSEASY, que provea al menos las operaciones mencionadas más arriba. Se debe explicar detalladamente qué información se guarda en cada parte, y las relaciones entre ellas.
- b) Resumir de qué manera se resolverían las operaciones listadas anteriormente, de acuerdo la estructura de representación elegida, justificando por qué se cumpliría la cota de complejidad solicitada, además de hacer las aclaraciones necesarias sobre *aliasing*.
- c) Escribir el algoritmo de las operaciones PUBLICARCURSO e INSCRIBIRCURSO y justificar de manera formal y detallada su complejidad en **peor y mejor caso**.
- d) Escribir de manera coloquial y formal el invariante de representación.
- e) Escribir de manera formal la función de abstracción.
- f) ¿Qué habría que modificar en la estructura y en la implementación de las operaciones vistas, de ser necesario, si se deseara implementar la siguiente operación? Justificar.
  - **CURSOSDISPONIBLES**(**in** sistema: courseasy)  $\rightarrow$  itACursos  
 {**Pre**: Ninguna.}

Obtiene los cursos del sistema con inscripción abierta y cupo disponible, ordenados por cupo disponible (descendente). Se debe retornar un iterador a una estructura que contenga cursos, con el único requisito de que los usuarios la puedan recorrer por completo en tiempo acotado estrictamente por  $f(c) = c \log(c)$ , donde  $c$  es la cantidad total de cursos disponibles.

Complejidad:  $O(1)$

Para la resolución del ejercicio no está permitido utilizar módulos implementados con tabla hash de base. Esto se debe a dos motivos: uno porque queremos que combinen el resto de las estructuras vistas, y otro porque los peores casos de la tabla hash exceden los que se piden en los ejercicios (por ejemplo, duplicar el espacio en una tabla cuesta  $O(n)$  en el peor caso).

Donde se menciona a **conjunto** en las definiciones de las operaciones anteriores se debe usar alguno de los módulos vistos que se explican con el CONJUNTO, de acuerdo a lo que está en el apunte de Módulos Básicos, o definir un módulo nuevo que también se explique con ese TAD.